Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №6**

**з дисципліни «Програмування паралельних комп’ютерних систем»**

**на тему «Java. Монітори»**

Виконав:

студент ІІI курсу

групи ІО-53

Лісовий В. О.

Перевірив:

доц. Корочкін О. В.

Київ – 2018

**Тема**

«Програмування для комп’ютерних систем зі спільною пам’яттю. Java. Монітори»

**Технічне завдання**

Розробити програму для розв’язання ПКС із СП (структура на рис. 1) математичної задачі: A = (B \* C) \* S + min(T) \* Z \* (MO \* MH).

Мова програмування: Java.

Засоби взаємодії: бар’єри, критичні секції, замки.



Рис. 1 Структурна схема ПКС

**Виконання роботи**

**Етап 1. Побудова паралельного алгоритму**

1. ai = (BH \* CH), i = 1..4
2. a = a + ai
3. bi = min(TH)
4. b = min(b, bi)
5. AH = a \* SH + b \* Z \* MO \* MHH;

Спільний ресурс: a, b, Z, MO.

**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |  |
| --- | --- | --- |
| № | Задача Т1 |  |
| 1 | Ввід даних (B, C, MH) |  |
| 2 | Сигнал усім потокам про завершення вводу даних | S\*,1 |
| 3 | Чекати завершення вводу даних в Т3, Т4 | W3,1 , W4,1 |
| 4 | Обчислення a1 = (BH \* CH) |  |
| 5 | Обчислення a = a + a1 | КД |
| 6 | Обчислення b1 = min (TH) |  |
| 7 | Обчислення b = min (b, b1) | КД |
| 8 | Копіювання al = a, b1 = b, Z1 = Z, MO1 = MO | КД |
| 9 | Сигнал усім потокам про завершення обчислення скаляру та пошуку мінімуму | S\*,2 |
| 10 | Чекати завершення завершення обчислення скаляру та пошуку мінімуму в Т2, Т3, Т4 | W\*,2 |
| 11 | Обчислення AH = a1 \* SH + b1 \* Z1 \* MO1 \* MHH |  |
| 12 | Чекати завершення завершення обчислення в Т2, Т3, Т4 | W\*,3 |
| 13 | Вивід А |  |

|  |  |  |
| --- | --- | --- |
| № | Задача Т2 |  |
| 1 | Чекати завершення вводу даних в Т1, Т3, Т4 | W\*,1 |
| 2 | Обчислення a2 = (BH \* CH) |  |
| 3 | Обчислення a = a + a2 | КД |
| 4 | Обчислення b2 = min (TH) |  |
| 5 | Обчислення b = min (b, b2) | КД |
| 6 | Копіювання a2 = a, b2 = b, Z2 = Z, MO2 = MO | КД |
| 7 | Сигнал усім потокам про завершення обчислення скаляру та пошуку мінімуму | S\*,2 |
| 8 | Чекати завершення завершення обчислення скаляру та пошуку мінімуму в Т1, Т3, Т4 | W\*,2 |
| 9 | Обчислення AH = a2 \* SH + b2 \* Z2 \* MO2 \* MHH |  |
| 10 | Сигнал T1 про завершення обчислення | S1,3 |

|  |  |  |
| --- | --- | --- |
| № | Задача Т3 |  |
| 1 | Ввід даних (S, MO) |  |
| 2 | Сигнал усім потокам про завершення вводу даних | S\*,1 |
| 3 | Чекати завершення вводу даних в Т1, Т4 | W1,1 , W4,1 |
| 4 | Обчислення a3 = (BH \* CH) |  |
| 5 | Обчислення a = a + a3 | КД |
| 6 | Обчислення b3 = min (TH) |  |
| 7 | Обчислення b = min (b, b3) | КД |
| 8 | Копіювання a3 = a, b3 = b, Z3 = Z, MO3 = MO | КД |
| 9 | Сигнал усім потокам про завершення обчислення скаляру та пошуку мінімуму | S\*,2 |
| 10 | Чекати завершення завершення обчислення скаляру та пошуку мінімуму в Т1, Т2, Т4 | W\*,2 |
| 11 | Обчислення AH = a3 \* SH + b3 \* Z3 \* MO3 \* MHH |  |
| 12 | Сигнал T1 про завершення обчислення | S1,3 |

|  |  |  |
| --- | --- | --- |
| № | Задача Т4 |  |
| 1 | Ввід даних (T, Z) |  |
| 2 | Сигнал усім потокам про завершення вводу даних | S\*,1 |
| 3 | Чекати завершення вводу даних в Т1, Т3 | W1,1 , W3,1 |
| 4 | Обчислення a4 = (BH \* CH) |  |
| 5 | Обчислення a = a + a4 | КД |
| 6 | Обчислення b4 = min (TH) |  |
| 7 | Обчислення b = min (b, b4) | КД |
| 8 | Копіювання a4 = a, b4 = b, Z4 = Z, MO4 = MO | КД |
| 9 | Сигнал усім потокам про завершення обчислення скаляру та пошуку мінімуму | S\*,2 |
| 10 | Чекати завершення завершення обчислення скаляру та пошуку мінімуму в Т1, Т2, Т3 | W\*,2 |
| 11 | Обчислення AH = a4 \* SH + b4 \* Z4 \* MO4 \* MHH |  |
| 12 | Сигнал T1 про завершення обчислення | S1,3 |

|  |  |  |
| --- | --- | --- |
| № | Задача Т5 |  |
| 1 | Чекати завершення вводу даних в Т1, Т3, Т4 | W\*,1 |
| 2 | Обчислення a5 = (BH \* CH) |  |
| 3 | Обчислення a = a + a5 | КД |
| 4 | Обчислення b5 = min (TH) |  |
| 5 | Обчислення b = min (b, b5) | КД |
| 6 | Копіювання a5 = a, b5 = b, Z5 = Z, MO5 = MO | КД |
| 7 | Сигнал усім потокам про завершення обчислення скаляру та пошуку мінімуму | S\*,2 |
| 8 | Чекати завершення завершення обчислення скаляру та пошуку мінімуму в Т1, Т3, Т4 | W\*,2 |
| 9 | Обчислення AH = a5 \* SH + b5 \* Z5 \* MO5 \* MHH |  |
| 10 | Сигнал T1 про завершення обчислення | S1,3 |

|  |  |  |
| --- | --- | --- |
| № | Задача Т6 |  |
| 1 | Чекати завершення вводу даних в Т1, Т3, Т4 | W\*,1 |
| 2 | Обчислення a6 = (BH \* CH) |  |
| 3 | Обчислення a = a + a6 | КД |
| 4 | Обчислення b6 = min (TH) |  |
| 5 | Обчислення b = min (b, b6) | КД |
| 6 | Копіювання a6 = a, b6 = b, Z6 = Z, MO6 = MO | КД |
| 7 | Сигнал усім потокам про завершення обчислення скаляру та пошуку мінімуму | S\*,2 |
| 8 | Чекати завершення завершення обчислення скаляру та пошуку мінімуму в Т1, Т3, Т4 | W\*,2 |
| 9 | Обчислення AH = a6 \* SH + b6 \* Z6 \* MO6 \* MHH |  |
| 10 | Сигнал T1 про завершення обчислення | S1,3 |

**Етап 3. Розробка програми**

package org.les;

import org.les.monitor.ResourceMonitor;

import org.les.monitor.SignalMonitor;

import org.les.tasks.MinTask;

import org.les.tasks.ScalarTask;

import org.les.threads.MyRunnable;

import java.util.concurrent.ForkJoinPool;

import java.util.concurrent.RecursiveTask;

public class Main {

public static int n = 2400;

public static int num = 1;

public static int p = 6;

public static int LIMIT = n / 8 + 1;

public static float h = (float) n / p;

private volatile static int[] A = new int[n], B = new int[n], C = new int[n], S = new int[n], T = new int[n];

private volatile static int[][] MH = new int[n][n];

private static ResourceMonitor resourceMonitor = new ResourceMonitor();

private static SignalMonitor signalMonitor = new SignalMonitor();

private static ForkJoinPool forkJoinPool = ForkJoinPool.commonPool();

public static void main(String[] args) throws InterruptedException {

Thread[] threads = new Thread[p];

for (int i = 0; i < threads.length; i++)

threads[i] = new Thread(new MyRunnable(i, A, B, C, S, T, MH, resourceMonitor, signalMonitor));

for (Thread thread : threads)

thread.start();

signalMonitor.waitInputSignal();

RecursiveTask<Integer> minTask = new MinTask(T);

forkJoinPool.execute(minTask);

resourceMonitor.setMin(minTask.join());

RecursiveTask<Integer> scalarTask = new ScalarTask(B, C);

forkJoinPool.execute(scalarTask);

resourceMonitor.setScalar(scalarTask.join());

forkJoinPool.shutdown();

signalMonitor.setMinScalarSignal();

for (Thread thread : threads)

thread.join();

}

}

package org.les.threads;

import org.les.Main;

import org.les.monitor.ResourceMonitor;

import org.les.monitor.SignalMonitor;

import org.les.utils.MathUtils;

import java.util.Arrays;

public class MyRunnable implements Runnable {

private int id;

private int scalar;

private int min;

private int[] A, B, C, S, T, Z;

private int[][] MO, MH;

private ResourceMonitor resourceMonitor;

private SignalMonitor signalMonitor;

public MyRunnable(int id, int[] a, int[] b, int[] c, int[] s, int[] t, int[][] MH,

ResourceMonitor resourceMonitor, SignalMonitor signalMonitor) {

this.id = id;

this.A = a;

this.B = b;

this.C = c;

this.S = s;

this.T = t;

this.MH = MH;

this.resourceMonitor = resourceMonitor;

this.signalMonitor = signalMonitor;

}

public MyRunnable(int id) {

this.id = id;

}

@Override

public void run() {

System.out.printf("Thread%d start%n", id);

try {

input();

signalMonitor.waitMinScalarSignal();

copy();

calc();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.printf("Thread%d finish%n", id);

}

private void input() throws InterruptedException {

switch (id) {

case 0:

Arrays.fill(B, Main.num);

signalMonitor.setInputSignal();

break;

case 2:

Arrays.fill(C, Main.num);

for (int i = 0; i < MH.length; i++)

Arrays.fill(MH[i], Main.num);

signalMonitor.setInputSignal();

break;

case 3:

Arrays.fill(S, Main.num);

Arrays.fill(T, Main.num);

Z = resourceMonitor.getZ();

Arrays.fill(Z, Main.num);

MO = resourceMonitor.getMO();

for (int i = 0; i < MO.length; i++)

Arrays.fill(MO[i], Main.num);

resourceMonitor.setZ(Z);

resourceMonitor.setMO(MO);

signalMonitor.setInputSignal();

break;

default:

signalMonitor.waitInputSignal();

}

}

private void copy() {

scalar = resourceMonitor.getScalar();

min = resourceMonitor.getMin();

Z = resourceMonitor.getZ();

MO = resourceMonitor.getMO();

}

private void calc() throws InterruptedException {

int[][] MBuf = new int[Main.n][Main.n];

int leftIndex = (int) (id \* Main.h);

int rightIndex = (int) ((id + 1) \* Main.h);

MathUtils.matrixMul(MO, MH, MBuf, leftIndex, rightIndex);

MathUtils.vectorMatrixMul(Z, MBuf, A, leftIndex, rightIndex);

for (int i = leftIndex; i < rightIndex; i++) {

A[i] = scalar \* S[i] + min \* A[i];

}

if (id == 0) {

signalMonitor.waitCalcSignal();

System.out.printf("Thread%d A=%s%n", id, Arrays.toString(A));

} else {

signalMonitor.setCalcSignal();

}

}

}

package org.les.monitor;

import org.les.Main;

public class ResourceMonitor {

private volatile int scalar;

private volatile int min;

private volatile int[] Z;

private volatile int[][] MO;

public ResourceMonitor() {

this.scalar = 0;

this.min = 0;

this.Z = new int[Main.n];

this.MO = new int[Main.n][Main.n];

}

public synchronized void setScalar(int scalar) {

this.scalar = scalar;

}

public synchronized void setMin(int min) {

this.min = min;

}

public synchronized void setZ(int[] Z) {

this.Z = Z;

}

public synchronized void setMO(int[][] MO) {

this.MO = MO;

}

public int getScalar() {

return scalar;

}

public int getMin() {

return min;

}

public int[] getZ() {

return Z;

}

public int[][] getMO() {

return MO;

}

}

package org.les.threads;

import org.les.Main;

import org.les.monitor.ResourceMonitor;

import org.les.monitor.SignalMonitor;

import org.les.utils.MathUtils;

import java.util.Arrays;

public class MyRunnable implements Runnable {

private int id;

private int scalar;

private int min;

private int[] A, B, C, S, T, Z;

private int[][] MO, MH;

private ResourceMonitor resourceMonitor;

private SignalMonitor signalMonitor;

public MyRunnable(int id, int[] a, int[] b, int[] c, int[] s, int[] t, int[][] MH,

ResourceMonitor resourceMonitor, SignalMonitor signalMonitor) {

this.id = id;

this.A = a;

this.B = b;

this.C = c;

this.S = s;

this.T = t;

this.MH = MH;

this.resourceMonitor = resourceMonitor;

this.signalMonitor = signalMonitor;

}

public MyRunnable(int id) {

this.id = id;

}

@Override

public void run() {

System.out.printf("Thread%d start%n", id);

try {

input();

signalMonitor.waitMinScalarSignal();

copy();

calc();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.printf("Thread%d finish%n", id);

}

private void input() throws InterruptedException {

switch (id) {

case 0:

Arrays.fill(B, Main.num);

signalMonitor.setInputSignal();

break;

case 2:

Arrays.fill(C, Main.num);

for (int i = 0; i < MH.length; i++)

Arrays.fill(MH[i], Main.num);

signalMonitor.setInputSignal();

break;

case 3:

Arrays.fill(S, Main.num);

Arrays.fill(T, Main.num);

Z = resourceMonitor.getZ();

Arrays.fill(Z, Main.num);

MO = resourceMonitor.getMO();

for (int i = 0; i < MO.length; i++)

Arrays.fill(MO[i], Main.num);

resourceMonitor.setZ(Z);

resourceMonitor.setMO(MO);

signalMonitor.setInputSignal();

break;

default:

signalMonitor.waitInputSignal();

}

}

private void copy() {

scalar = resourceMonitor.getScalar();

min = resourceMonitor.getMin();

Z = resourceMonitor.getZ();

MO = resourceMonitor.getMO();

}

private void calc() throws InterruptedException {

int[][] MBuf = new int[Main.n][Main.n];

int leftIndex = (int) (id \* Main.h);

int rightIndex = (int) ((id + 1) \* Main.h);

MathUtils.matrixMul(MO, MH, MBuf, leftIndex, rightIndex);

MathUtils.vectorMatrixMul(Z, MBuf, A, leftIndex, rightIndex);

for (int i = leftIndex; i < rightIndex; i++) {

A[i] = scalar \* S[i] + min \* A[i];

}

if (id == 0) {

signalMonitor.waitCalcSignal();

System.out.printf("Thread%d A=%s%n", id, Arrays.toString(A));

} else {

signalMonitor.setCalcSignal();

}

}

}